APPENDIX 1

```
%%%
function B = Initcluster(X,m)
% Get an initial base codebook B at random from input data

% INPUTS
% X = input data: each column is a RGB 'vector'
% m = number of base codevectors

% OUTPUTS
% B = base codevector matix

%need to duplicate some columns of B if X is small
[n,N] = size(X);
if(N > m)
 replace = 0;
else
 replace = 1;
end

%track what inputs put in B, so no duplication for X large
chosen = zeros(1,N);
B = zeros(n,m);
for i=1:m
 draw = floor(N*rand + 1);
 if(~replace)
  while(chosen(draw))
   draw = floor(N*rand + 1);
  end
```

```
    end
    B(:,i) = X(:,draw);
    chosen(draw) = 1;
  end

%%%

function [B,Bq,d,iters,rd,nX] = lgbo(X,m,base)

% INPUTS
% X = input data: each column is a RGB 'vector'
% m = number of codevectors (columns in C)
% base = number of base codevectors (columns of B)

% OUTPUTS
% B = base codevector matrix
% Bq = quantized base codevector matrix
% d = distortion of X when replaced with chosen codevectors
% iters = # of iterations to reach convergence

[d,N] = size(X);    % d = dimension, N = blocksize

%random initialization
Init = initcluster(X,base);     % choose random initial set

if base==2      %1D (linear interpolation)
        W = [1:(-1/(m-1)):0; 0:(1/(m-1)):1]; %weight matrix
elseif base==3 %2D
        W=[1,0,0;0,1,0;.25,.25,.5;-.25,-.25,1.5; ...
          1,-1,1;-1,1,1;0,-1,2;-1,0,2;]';
```

```
elseif base==4 %3D
        W=[eye(4),.25*[2,2,1,-1;2,-1,2,1;1,2,-1,2;-1,1,2,2]];
   %W=[eye(4),.125*[3,2,2,1;2,3,2,1;3,1,2,2;1,2,3,2]];
end
5
%variable initialization
stoppingeps = 1.e-5;
vi = ones(1,m);
index=zeros(1,N);mind=index;
10 cumdist = Inf;  lastdist = 0;

C=Init*W; %interpolate codevectors

# iterate until convergence
15 while(abs(cumdist - lastdist) > stoppingeps)
  lastdist = cumdist;
  cumdist = 0;
  iters=iters+1;

20   while(1) %1 iteration that repeats if B goes singular
    % step (A)
    % form Voronoi regions: for each input,
        % determine which centroid it is closest to
        for i=1:N
25     V=X(:,i(vi))-C;
      nm=sum(V.*V);  %Euclidean distance squared (MSE)
      %input i's closest codevector
      [mind(i),index(i)]=min(nm);
    end
30     cumdist = sum(mind);
```

```
%find diagonal matrix N
for j=1:m
    n(j) = sum(index==j);
end

% check if B is singular
% and force it to be non-singular
num=sum(n~=0);
if num<base  %fewer than base non-zero!
 nz = find(n);
     if (base-num)==1 %base==2
  [jy,ji]=max(mind);
  Init(:,1:(base-1))=B(:,nz);
  Init(:,base)=X(:,ji);
 else
  [jy,ji]=sort(mind);
  Init(:,1:num)=B(:,nz);
  jl=N;
  for jk=(num+1):base
    Init(:,jk)=X(:,ji(jl));
    while(sum(abs(X(:,ji(jl))-X(:,ji(jl-1))))==0)
      jl=jl-1;
    end
    jl=jl-1;
  end % for jk
 end  % else
     C=Init*W; %start all over if hit a singular matrix
else
 break;
```

```
 end % if not singular then end while loop
end %while(1)

s = zeros(d,m); %get sum matrix S
for j=1:m
      if n(j)==1
      s(:,j) = X(:,index==j);
  elseif n(j)
     s(:,j)=sum(X(:,index==j)')';
  end
end %for j=1:m

Init = s*W'*inv(W*diag(n)*W');  %new base codevector matrix B
C=Init*W; %new codevector matrix C

end %% while not converged

nX = B(:,index);        % save re-constructed block
d  = cumdist;           % total block distortion
      end % function
```